METHOD AND APPARATUS FOR THE AUTOMATIC DISCOVERY OF THE RELATIONSHIPS BETWEEN APPLICATIONS AND THEIR ASSOCIATED DATA AND CONFIGURATION FILES

CROSS REFERENCE TO RELATED APPLICATIONS

The present invention is related to *Method and Apparatus for the Automatic Migration of Applications and Their Associated Data and Configuration Files*, serial no. 09/_____, attorney docket no. YOR920010276US1, and *Method and Apparatus for Performing the Identification of Files to be Backed Up Using Relational Meta Data,* serial no. 09/_____, attorney docket no. YOR920010343US1 filed even date hereof, assigned to the same assignee, and incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates generally to an improved data processing system, and in particular to a method and apparatus for managing information about applications and files. Still more particularly, the present invention provides a method, apparatus, and computer implemented instructions for performing automatic discovery of relationships between applications and associated data for those applications.

2. Description of Related Art:

When an application is installed on a computer system, the setup program, which is part of the application and supplied by the application supplier, registers with the

operating system that the application will be creating data files of a certain file type on the file system. More specifically, the setup program tells the operating system the type of data files that will be created, edited, deleted, or otherwise manipulated by the application. Data files manipulated by the application are associated with that application as long as that application remains installed on the computer system, providing the file type is not changed. If the application is removed, the mapping of the file extension to the application is lost. The current state of the art is to identify the data file type with a three letter filename extension. For example, the setup program for Microsoft Word registers with the operating system to indicate that filenames with the extension ".doc" are associated with Microsoft Word. Microsoft Word is a word processing program available from Microsoft Corporation. As a result, when a user selects a data file to be opened, the operating system uses the filename extension of that data file to determine which application should be launched.

Users typically install dozens of applications on a data processing system, such as editors, e-mail programs, Internet browsers, and business applications such as payroll or inventory access. Each of these applications usually create, edit, rename, or delete data files of a certain data type as determined by the file type extension. These data files may include user identification such as users Ids and passwords, application configuration files, reports, and data files. Currently, each application consistently creates data file types associated with an application vendor determined filename extension. For example, Microsoft Word creates document data files using the filename extension ".doc", and Lotus WordPro creates document data files using the filename extension ".lwp", Lotus Notes creates data files using the filename extension ".id" for userid information and ".nsf" for notes databases. Lotus Word Pro and Lotus Notes are available from Lotus Development Corporation. Each application vendor selects a unique filename extension to be used for their application and hopes that no other application vendor selects the

same extension. In the case of some programs that use common types of files such as .mid or .jpg, installing another program will often incorrectly associate the data files with the newly installed application.

As the end user executes applications and works on data files, the list of data files associated with the application continually changes. Some data files may be created, others deleted or edited, and some renamed to other filenames or file type extensions. The user also may place some of these files in other places within the file system. With these types of file manipulations, the files may become hard to locate without using a tool to search the hard disk drive for filenames with a well known filename extension for a particular application. The files also may contain extensions that do not follow the recommended file naming convention. For example, the user may create a Microsoft Word document and save it with the filename extension of ".zoo" instead of the standard .doc extension.

Therefore, it would be advantageous to have an improved method and apparatus for tracking data and configuration files associated with these applications.

SUMMARY OF THE INVENTION

The present invention provides for a method, apparatus, and computer implemented instructions for tracking relationships between programs and data in a data processing system. A file access request is received from a program, wherein the request is received at an operating system level. An association between the application and data and configuration files is created and stored in a database that can be subsequently accessed.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a pictorial representation of a data processing system in which the present invention may be implemented in accordance with a preferred embodiment of the present invention;

Figure 2 is a block diagram of a data processing system in which the present invention may be implemented;

Figure 3 is a diagram illustrating components involved in automatic discovery of relationships between application programs and associated data in accordance with a preferred embodiment of the present invention;

Figure 4 is a diagram of meta data describing relationships between applications and associated data in accordance with a preferred embodiment of the present invention;

Figure 5 is a diagram illustrating an example call in accordance with a preferred embodiment of the present invention;

Figure 6 is a flowchart of a process used for installing the processes for automatically discovering relationships between applications and associated data in accordance with a preferred embodiment of the present invention;

Figure 7 is a flowchart of a process used for handling requests for file operations in accordance with a preferred embodiment of the present invention;

Figure 8 is a flowchart of a process used for processing an open operation in accordance with a preferred embodiment of the present invention;

Figure 9 is a flowchart of a process used for processing a delete operation in accordance with a preferred embodiment of the present invention;

Figure 10 is a flowchart of a process used for renaming in accordance with a preferred embodiment of the present invention;

Figure 11 is a flowchart of a process used for processing a close or copy operation in accordance with a preferred embodiment of the present invention;

Figure 12 is a flowchart of a process used for processing queries for file information in accordance with a preferred embodiment of the present invention; and

Figure 13 is a flowchart of a process used by an application to obtain a list of files in accordance with a preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

With reference now to the figures and in particular with reference to **Figure 1**, a pictorial representation of a data processing system in which the present invention may be implemented is depicted in accordance with a preferred embodiment of the present invention. Computer **100** is depicted which includes system unit **102**, video display terminal **104**, keyboard **106**, storage devices **108**, which may include floppy drives and other types of permanent and removable storage media, and mouse **110**. Additional input devices may be included with personal computer **100**, such as, for example, a joystick, touchpad, touch screen, trackball, microphone, and the like. Computer **100** can be implemented using any suitable computer, such as an IBM RS/6000 computer or IntelliStation computer, which are products of International Business Machines Corporation, located in Armonk, New York. Although the depicted representation shows a computer, other embodiments of the present invention may be implemented in other types of data processing systems, such as a network computer. Computer **100** also preferably includes a graphical user interface that may be implemented by means of systems software residing in computer readable media in operation within computer **100**.

With reference now to **Figure 2**, a block diagram of a data processing system is shown in which the present invention may be implemented. Data processing system **200** is an example of a computer, such as computer **100** in **Figure 1**, in which code or instructions implementing the processes of the present invention may be located. Data processing system **200** employs a peripheral component interconnect (PCI) local bus architecture. Although the depicted example employs a PCI bus, other bus architectures such as Accelerated Graphics Port (AGP) and Industry Standard Architecture (ISA) may be used. Processor **202** and main memory **204** are connected to PCI local bus **206** through PCI

bridge **208**. PCI bridge **208** also may include an integrated memory controller and cache memory for processor **202**. Additional connections to PCI local bus **206** may be made through direct component interconnection or through add-in boards. In the depicted example, local area network (LAN) adapter **210**, small computer system interface (SCSI) host bus adapter **212**, and expansion bus interface **214** are connected to PCI local bus **206** by direct component connection. In contrast, audio adapter **216**, graphics adapter **218**, and audio/video adapter **219** are connected to PCI local bus **206** by add-in boards inserted into expansion slots. Expansion bus interface **214** provides a connection for a keyboard and mouse adapter **220**, modem **222**, and additional memory **224**. SCSI host bus adapter **212** provides a connection for hard disk drive **226**, tape drive **228**, and CD-ROM drive **230**. Typical PCI local bus implementations will support three or four PCI expansion slots or add-in connectors.

An operating system runs on processor **202** and is used to coordinate and provide control of various components within data processing system **200** in **Figure 2**. The operating system may be a commercially available operating system such as Windows 2000, which is available from Microsoft Corporation. An object oriented programming system such as Java may run in conjunction with the operating system and provides calls to the operating system from Java programs or applications executing on data processing system **200**. "Java" is a trademark of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented programming system, and applications or programs are located on storage devices, such as hard disk drive **226**, and may be loaded into main memory **204** for execution by processor **202.**

Those of ordinary skill in the art will appreciate that the hardware in **Figure 2** may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash ROM (or equivalent nonvolatile memory) or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in **Figure 2**.

Also, the processes of the present invention may be applied to a multiprocessor data processing system.

For example, data processing system **200**, if optionally configured as a network computer, may not include SCSI host bus adapter **212**, hard disk drive **226**, tape drive **228**, and CD-ROM **230**, as noted by dotted line **232** in **Figure 2** denoting optional inclusion. In that case, the computer, to be properly called a client computer, must include some type of network communication interface, such as LAN adapter **210**, modem **222**, or the like. As another example, data processing system **200** may be a stand-alone system configured to be bootable without relying on some type of network communication interface, whether or not data processing system **200** comprises some type of network communication interface. As a further example, data processing system **200** may be a personal digital assistant (PDA), which is configured with ROM and/or flash ROM to provide nonvolatile memory for storing operating system files and/or user-generated data.

The depicted example in **Figure 2** and above-described examples are not meant to imply architectural limitations. For example, data processing system **200** also may be a notebook computer or hand held computer in addition to taking the form of a PDA. Data processing system **200** also may be a kiosk or a Web appliance.

The processes of the present invention are performed by processor **202** using computer implemented instructions, which may be located in a memory such as, for example, main memory **204**, memory **224**, or in one or more peripheral devices **226-230**.

The present invention provides a method, apparatus, and computer implemented instructions for performing the automatic discovery of the relationships between application programs and their associated data and configuration files. In the depicted example, the mechanism of the present invention includes a software program and software device driver mechanism that is installed on the computer system at the time the

operating system is installed. These components may be installed at a later time, but the discovery of the relational meta data only begins when components of the present invention have been installed. The software program "hooks" or connects to the operating system at the single point where all file-oriented requests are handled. The single point of entry for various classes of operating system services is a standard feature of all currently available operating systems. When any executing program (application, service, etc.) makes a request to open, close, delete, rename, or move a file, the request is detected, and the name of the requesting program is identified The being operated on and the name of the program accessing the file is used to automatically create a relationship between the two. This relationship, file and program, is captured and represented in a relational meta data format. Additional meta data about the file creation can also be captured, such as the location of the file, time, date, or identity of the user. This relational meta data can be stored in another data file in the file system or saved in a database (i.e. registry, database, directory, etc.). The database can be protected and hidden from users to prevent the deletion or corruption of the data.

With reference now to **Figure 3**, a diagram illustrating components involved in automatic discovery of relationships between application programs and associated data is depicted in accordance with a preferred embodiment of the present invention. The components illustrated in **Figure 3** may be implemented as software and data structures in a data processing system, such as data processing system **200** in **Figure 2**.

In the depicted examples, operating system **300** includes file functions **302**. These files functions are used to perform different functions on files, such as file **304** in response to requests from applications, such as application **306**. These functions include, for example, opening, closing, creating, copying, renaming, and deleting files. When the user starts application **306**, application **306** generally requires a data file to act upon. For instance, starting a word processor usually requires that the user indicate the name of the

file to be created, edited or processed. Most applications have some type of open menu where the user specifies which file they are going to work on. The user generally clicks a "file open" button or menu item to open a file, and is then presented with a list of files meeting that criteria to work on.

Using current technology, the list of files available to work on is determined the file type, which may be identified through the file type extension. When an application is installed, it usually notifies the operating system, which file type extension should be associated with that program. For example, Microsoft Word notifies the operating system that it will use files with the .doc extension. After the application is installed, if the user selects a file with the .doc extension, the Microsoft Word application will be launched to operate on that file. Using current technology, the association between the application program and the file type extension exists until that application program is removed from the system. When the application is removed, the removal program also removes any associations that had been established at the time the application was installed.

With the mechanism of the present invention, calls by application **306** to file functions **302** are hooked or routed to device driver **308**. These functions calls include opening, closing, creating, copying, renaming, and deleting a file. Each time a call for one of the file functions is made, the call is intercepted by device driver **308**. The name of the program making the call is identified by device driver/service **308** along with the name of the data file being operated on.

For example, device driver **308** hooks the single entry point of the "file close" function. Each time a file, such as file **304**, is closed, the close is intercepted by device driver **308**. This device driver identifies the name of application **304** closing file **304**, along with the name of file **304**. In this example, file **304** is opened and closed by application **306**, representing a normal close of file **304**. The relational meta data that represents the association of file **304** to application **306** is updated in database **310** with

the new information. If application **306** opens file **304**, but another software entity, such as operating system **300** closes file **304**, then an abnormal close may have occurred because of a failure in application **306**.

Each time a file is opened or closed, the relational meta data for the given file is updated by device driver **308**. The mechanism of the present invention also may hook the operating system entry points for file erase, file rename, file move, and file copy functions at the device driver level or at the operating system service level. These additional hooks also update the relational meta data in database **310**. If an application program, in the process of executing, creates a file, the file creation information and association to the application program is stored in relational meta data. If the application program deletes a file, the relational meta data for the deleted file is deleted. The relational meta data for file **304** is updated in database **310** and is updated if the application renames file **304**. It is important to note that, in these examples, in the event that the same file is accessed by more than one program, the database will contain the reference to the application that accessed the file most recently.

If the user copies file **304** to another location, the relational meta data for file **304** is updated with the new location. If multiple applications perform activity against file **304**, the relational meta data for file **304** is updated to reflect the association to multiple applications.

When the application **306** is started, the user is presented with a list of files to work on, depending on the file type extension registered with the operating system by application **306**. The user selects one or more files to work on, and then confirms the choice by clicking an OK button or similar type of control. Some application programs, such as Microsoft Word, keep a finite length list of the files acted upon in persistent storage. One of the options the user has is to select the history list of files that have been accessed by the application program. The mechanism of the present invention provides a

method, apparatus, and computer implemented instructions for a convenient way to provide quick access to frequently manipulated files.

The list of files displayed that can be acted upon is based on the file type extension. However, the user may have renamed the file with a different extension, or moved the file to another area on the disk or even another computer or network share. Application **304** has no direct knowledge of these files, their new extension, or their new location because the file type extension has changed or the files have been moved to an unknown location. Because this information is in database **301**, application **304** can query database **310** through calls to device driver **308** to find the file names and location of all of the data and configuration files associated with the application **304**. Application **304** then uses the list of files from database **310** to present to the user at the time application **304** is run. Instead of choosing a data file of a certain file type extension and from a specified physical location on the disk, the user can now select any file that had been created by application **304** and from any location on the disk. The access to database **310** may be provide through standard application programming interface (API) calls made to device driver **308** from application **304** or another application. Using the access methods provided by the invention the user can query the relational database with such queries as:

Show me the files created between December 1, 2000 and December 15, 2000.

Show me the files created since January 1, 2001 by the user stevemas.

Show me all of the files associated to the Lotus 1-2-3 program.

The association of applications with files and file locations may extend to files created, stored, or moved on remote storage devices located on another computer system. The mechanism of the present invention may be installed as an integral part of operating system **300**, such as within a kernel. Alternatively, the mechanism may be added as a patch or add-on component if added to operating system **300** after its installation.

Turning next to **Figure 4**, a diagram of meta data describing relationships

between applications and associated data is depicted in accordance with a preferred embodiment of the present invention. In the depicted example, records **400**, **402**, and **404** are examples of meta data, which may be stored in a database, such as database **310** in **Figure 3**. Record **400** includes sections **406**, **408**, **410**, **412**, **414**, and **416**. Section **406** identifies the date of the last file update. Section **408** indicates the last time the file was access in hours, minutes, and seconds. Section **410** identifies the name of the file, while section **412** identifies the location of the file. The application making the call for the function is identified in section **414**. The user is identified in section **416**.

With reference now to **Figure 5**, a diagram illustrating an example call is depicted in accordance with a preferred embodiment of the present invention. Call **500** is an example of a call, which may be used to obtain a file list. The call specifies an application name as well as criteria, which may be used to search for records, such as records **400**, **402**, and **404** in **Figure 4** within database **310** in **Figure 3**. The criteria may be, for example, a list of files associated with the Lotus Word Pro application that are more than 30 days old.

Turning next to **Figure 6**, a flowchart of a process used for installing the processes for automatically discovering relationships between applications and associated data is depicted in accordance with a preferred embodiment of the present invention. The process begins by detecting a system boot of the data processing system (step **600**). Next, hooks are installed (step **602**). The hooks installed are those for use by a device driver, such as device driver **308** in **Figure 3** to hook or intercept calls for file functions. Then, the system boot is continued (step **604**) with the process terminating thereafter.

The flowcharts illustrated in **Figures 7-11** are examples of processes used to automatically discover relationships between applications and associated data. With reference now to **Figure 7**, a flowchart of a process used for handling requests for file operations is depicted in accordance with a preferred embodiment of the present

invention. The process illustrated in **Figure 7** may be implemented in a device driver, such as device driver **308** in **Figure 3**.

The process begins by receiving a request for a file operation (step **700**). Next, a determination is made as to whether the file operation is to open a file (step **702**). If the file operation is not open, then a determination is made as to whether the file is to be deleted (step **704**). If the file is not to be deleted, a determination is made as to whether the file is to be renamed (step **706**).

If the file is not to be renamed, a determination is made as to whether the file is closed or copied (step **708**). If the file is not to be closed or copied, file operation continues (step **710**) with the process terminating thereafter. At this point, the file operation request is passed to the actual file function that is to process the request.

With reference again to step **708**, if the file is to be closed or copied, close or copy operation is performed (step **712**) with the process proceeding to step **710**. Turning back to step **706**, if the file is to be renamed, a rename operation is performed (step **714**) with the process proceeding to step **710** thereafter. With reference again to step **704**, if the file is to be deleted, a delete operation is performed (step **716**) and the process proceeds to step **710** as described above. With reference again to step **702**, if the file is opened, an open operation is performed (step **718**) with the process proceeding to step **710**.

Turning next to **Figure 8**, a flowchart of a process used for processing an open operation is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 8** is a more detailed description of step **718** in **Figure 7**.

This process is called in response to an open operation being present. The process begins with a determination as to whether a record of the file identified for the operation is present in the database (step **800**). If the file is not present in the database, an identification of the file is added to the database (step **802**) with the process returning thereafter for a continuation of the file operation. The identification may include, for

example, the name of the file, the name of application requesting the operation, a date, and a time of the request.

Otherwise, a determination is made as to whether the file is found in the same location (step **804**). If the file is found at the same location, the process returns to continue process the file operation. If the file is not in the same location, the record is updated with the new location (step **806**) with the process then returning to continue processing of the file operation. The open operation occurs immediately because the database cannot be updated until it is known that the file can be opened.

With reference now to **Figure 9**, a flowchart of a process used for processing a delete operation is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 9** is a more detailed description of step **716** in **Figure 7**.

The process begins with a determination as to whether a record of the file is in a database (step **900**). If the file is in the database, the database is updated (step **902**) with the process then returning to continue the file operation. This update reflects the application closing the file as well as other information, such as a time and date of the operation. Otherwise, the process returns without performing any action in the database. In this instance, the file is not tracked by the mechanism of the present invention.

Turning next to **Figure 10**, a flowchart of a process used for renaming is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 10** is a more detailed description of step **714** in **Figure 7**.

The process begins with a determination as to whether a record of the file is in the database (step **1000**). If the file is not in the database, a new record is established in the database (step **1002**), and the process returns to continue processing the file operation. The new record may be in a format, such as, for example, record **400** in **Figure 4**. Otherwise, the database is updated (step **1004**) with the process returning for continued

processing of the file operation.

With reference now to **Figure 11**, a flowchart of a process used for processing a close or copy operation is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 11** is a more detailed description of step **712** in **Figure 7**.

The process begins with a determination as to whether a record of the file is in a database (step **1100**). If the file is in the database, a reference is updated (step **1102**) with the process returning to continue the file operation. Otherwise, a new record for the file is added to the database (step **1104**), and the process returns for continuation of the file operation.

Turning next to **Figure 12**, a flowchart of a process used for processing queries for file information is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 12** may be implemented in a device driver, such as device driver **308** in **Figure 3**.

The process begins by receiving a call for file information (step **1200**). This call may be received from an application, such as application **306** in **Figure 3**. Next, a database is queried for file names and locations of files for the application identified in the query (step **1202**). A result is received from the database (step **1204**), and returned to the caller (step **1206**) with the process terminating thereafter.

With reference now to **Figure 13**, a flowchart of a process used by an application to obtain a list of files is depicted in accordance with a preferred embodiment of the present invention. The process illustrated in **Figure 13** may be implemented in an application, such as application **304** in **Figure 3**.

The process begins by sending a call for a list of files for an application (step **1300**). This application may be the application generating the call for the list or for another application. This call is sent to a device driver, such as device driver **308** in

Figure 3. Next, a result is received (step **1302**). Then, a list of file names and locations is presented on a display to the user (step **1304**) with the process terminating thereafter.

One use of the mechanism of the present invention is for migration of applications. When the user needs to migrate the application file data to another system, the user first installs the relevant applications on the new system. The user then invokes the invention and selects the "migrate" option. This option presents a menu of the files to migrate by reading the relational meta data for each file that is related to the relevant applications. The user can then accept or refuse for one or more files to be copied or migrated to the new system.

Thus, the present invention provides an improved method, apparatus, and computer implemented instructions for dynamically discovering relationships between applications and associated data. This mechanism provides an advantage in identifying files for a particular application at a later time. Files that are created, renamed, moved, copied, and deleted may be identified if needed. The mechanism employs identifying this data at the point at which calls are made for file functions. The meta data is stored in a data structure, such as a database, in these examples. Of course other types of data structure may be used depending on the particular implementation.

It is important to note that while the present invention has been described in the context of a fully functioning data processing system, those of ordinary skill in the art will appreciate that the processes of the present invention are capable of being distributed in the form of a computer readable medium of instructions and a variety of forms and that the present invention applies equally regardless of the particular type of signal bearing media actually used to carry out the distribution. Examples of computer readable media include recordable-type media, such as a floppy disk, a hard disk drive, a RAM, CD-ROMs, DVD-ROMs, and transmission-type media, such as digital and analog communications links, wired or wireless communications links using transmission forms,

such as, for example, radio frequency and light wave transmissions. The computer readable media may take the form of coded formats that are decoded for actual use in a particular data processing system.

The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. Although the depicted illustrations show the mechanism of the present invention embodied on a single server, this mechanism may be distributed through multiple data processing systems. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.